PRS User Manual

PRS is developed to run in a parallel computing environment based on a single machine with multiple processors. This User Manual provides detailed instructions for setting up PRS to solve large-scale R&S problems on Windows systems and Debian-based Linux systems, such as Debian, Ubuntu, and Linux Mint. It consists of the following sections:

1 Installing Necessary Software Tools

To run PRS, users must first install Apache Spark, Java Development Kit (JDK), and Python. In particularly, we use Apache Spark version 3.4.0, JDK version 1.8, and Python version 3.10.x to ensure compatibility and stability. The following subsections provide detailed instructions for installing these software tools on Windows systems and Debian-based Linux systems.

1.1 Installing Software Tools on Windows

1.1.1 Installing Apache Spark 3.4.0

A. Downloading

To obtain Apache Spark 3.4.0, users can visit https://archive.apache.org/dist/spark/spark-3.4.0/ and download the `spark-3.4.0-bin-hadoop3.tgz` file.

B. Installing

To install Apache Spark 3.4.0, users only need to extract the `spark-3.4.0-bin-hadoop3.tgz` file. Upon extraction, a folder, named `spark-3.4.0-bin-hadoop3`, will be created, containing all the extracted files and subfolders. It is crucial for the users to rename the `spark-3.4.0-bin-hadoop3` folder to `SparkPRS`.

1.1.2 Installing JDK 1.8

A. Downloading

- Visit the downloads page https://www.oracle.com/java/technologies/downloads/?er=221886.
- Scroll down to find **Java 8** and choose **Windows**.
- Download the appropriate version of installer, e.g., `jdk-8u431-windows-x64.exe` for Windows 64-bit operating system and `jdk-8u431-windows-i586.exe` for Windows 32-bit operating system. Notice that users need to have an Oracle account or sign in if prompted, as Oracle may require authentication to download older versions.

B. Installing

- Locate the installer file and double-click it to start the installation.
- Follow the on-screen instructions and choose default settings to complete the installation process.

1.1.3 Installing Python 3.10.x

A. Downloading

- Visit the downloads page https://www.python.org/downloads/.
- Scroll down to find the release version **3.10.x**, e.g., **3.10.11**, and click **Download**.
- Download the appropriate version of installer, e.g., Windows installer (64-bit) or Windows installer (32-bit).

B. Installing

- Locate the installer file, i.e., Windows installer (64-bit) or Windows installer (32-bit), and double-click it to start the installation.
- Check the option "**Add Python to PATH**" and click **Install Now** to proceed with the default installation.
- Wait for the installation to complete, then click **Close**.

C. Adding Packages

To run PRS, users need to install following Python packages: `numpy`, `pyspark`, `numba`, and `scipy`. In what follows, we provide an example of installing these packages on Windows systems.

- Press **Win + R**, type **cmd**, and press **Enter** to open Command Prompt.
- Check if `pip` is installed by running the `pip –version` command. If `pip` is not recognized, users may reinstall Python and ensure that the option "**Add Python to PATH**" is selected during installation.
- Execute the `py -3.10 -m pip install <package_name>` command to install each package.

1.2 Installing Software Tools on Linux

Before installing the software tools on a Linux system, users should execute the `sudo apt-get update` command in the terminal to fetch the latest information about available software packages from configured software repositories.

1.2.1 Installing Apache Spark 3.4.0

A. Downloading

Download the software package by executing the following command in the terminal:

```
wget https://archive.apache.org/dist/spark/spark-3.4.0/spark-3.4.0-bin-hadoop3.tgz
```

B. Installing

Extract the downloaded package to Desktop and rename it to `SparkPRS` by executing the following commands in the terminal:

```
tar -xvzf spark-3.4.0-bin-hadoop3.tgz -C ~/Desktop
mv ~/Desktop/spark-3.4.0-bin-hadoop3 ~/Desktop/SparkPRS
```

1.2.2 Installing JDK 1.8

Execute the `sudo apt-get install openjdk-8-jdk` command in the terminal.

1.2.3 Installing Python 3.10.x

A. Installing

Execute the `sudo apt-get install python3.10 python3.10-tk python3-pip` command in the terminal.

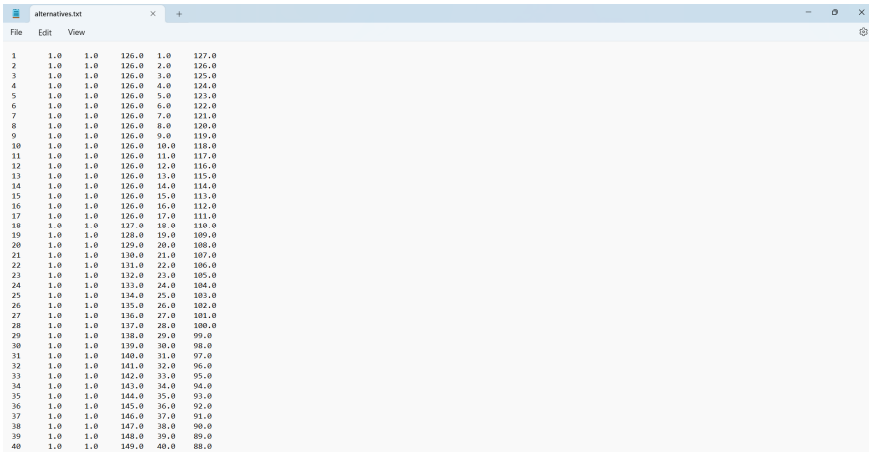
B. Adding Packages

As with the installation process on the Windows system, users are required to install the following Python packages: `numpy`, `pyspark`, `numba`, and `scipy`. To install these packages, users can open the terminal and execute the `python3.10 -m pip install <package_name>` command to install each package.

2 Creating the `alternatives.txt` File

To apply PRS to solve a specific large-scale R&S problem, users must create a text file, named `alternatives.txt`, which contains the alternative information for the problem. The file must follow some specific formatting rules. Each line in the file corresponds to a distinct alternative and consists of multiple entries separated by spaces. These entries are all of numerical data type. The first entry specifies the index of the alternative, while the subsequent entries record the values for the parameters that define each alternative. For example, in a throughput maximization problem, to define an alternative, one needs to set values for five parameters, namely, s_1, s_2, s_3, b_2, and b_3. Consequently, each line in the text file should include six entries: the first indicating the alternative's index, followed by five entries representing the values of s_1, s_2, s_3, b_2, and b_3 in that order. Notice that the order in which each parameter is recorded in each line affects how we assign values to different parameters when defining methods or functions to

generate simulation samples. The screenshot below provides an example of the `alternatives.txt` file created for the throughput maximization problem with 1,016,127 alternatives considered in the paper.



3 Setting up PRS

In the following subsections, we separately discuss the remaining setup processes for the Java and Python versions of PRS.

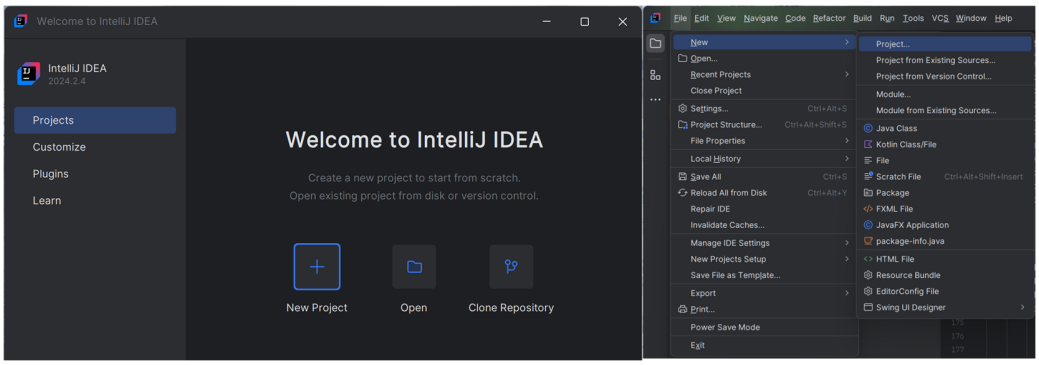
3.1 Setting up the Java Version of PRS

To set up the Java version of PRS, users may need an integrated development environment (IDE) to deploy, edit, and package the source code effectively. It is recommended to use IntelliJ IDEA (Community Edition) to perform these tasks. Users can download the IDE from the official website for IntelliJ IDEA. The remaining setup process for the Java version of PRS consists of four steps: 1) Creating a Maven project in IntelliJ IDEA; 2) Deploying the source code to the Maven project; 3) Packaging the Maven project as a JAR file; and 4) Running PRS. In what follows, we discuss these four steps in details.

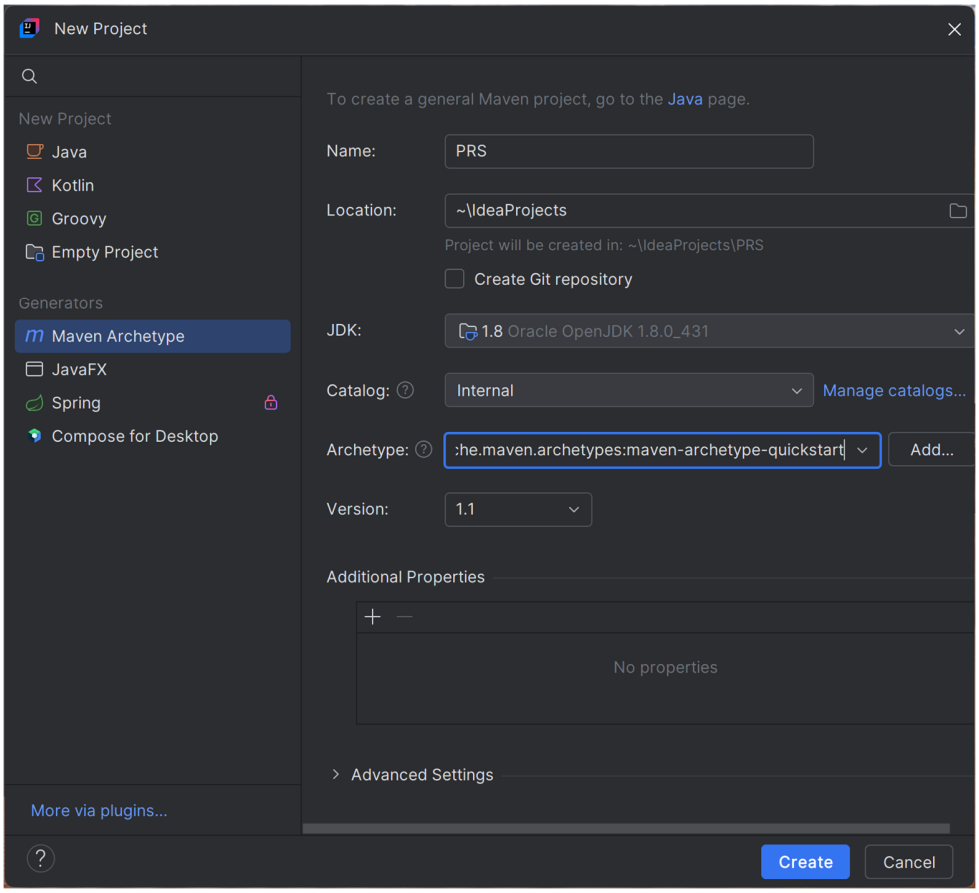
3.1.1 Creating a Maven Project in IntelliJ IDEA

A. Creating a New Project in IntelliJ IDEA

- Launch IntelliJ IDEA.
- Select **New Project** from the welcome screen or **File** → **New** → **Project** if users are already inside IntelliJ IDEA.

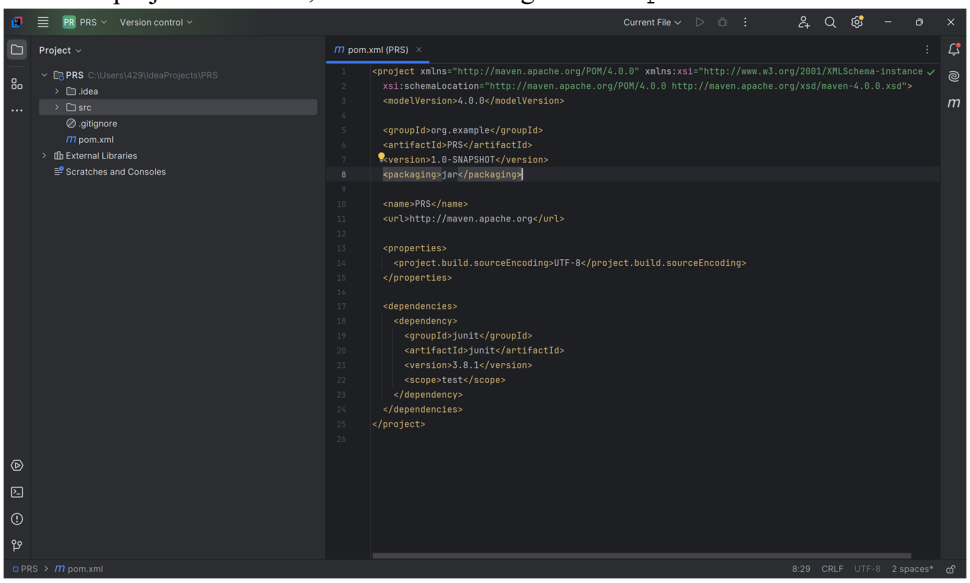


- In the New Project wizard, select **Maven Archetype** on the left-hand side. Then set the project name, e.g., PRS, select **JDK 1.8** for the project, choose **archetype maven-archetype-quickstart**, and click **Create**.



B. Adding Dependencies and Build Plugins in the `pom.xml` File:

- Once the project is created, users should navigate to the `pom.xml` file.



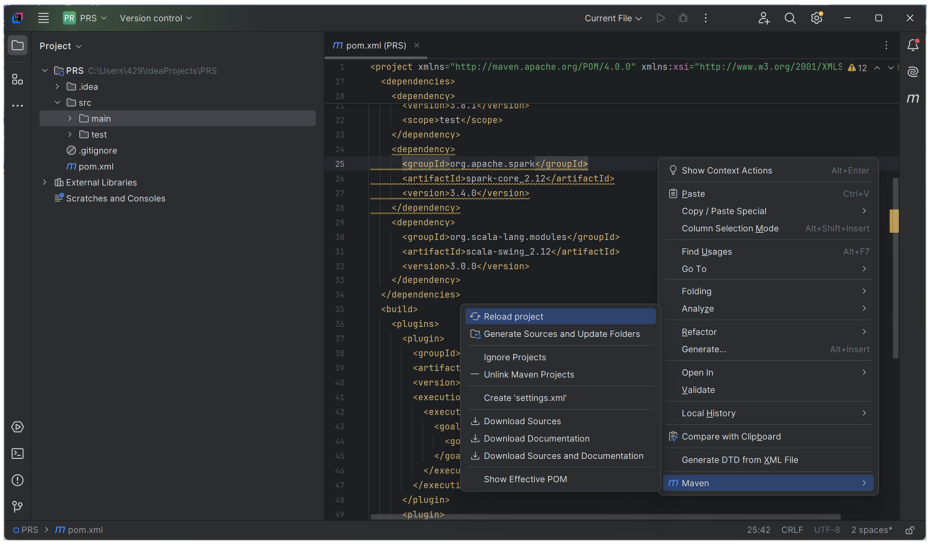
- Add the **spark-core 2.12** and **scala-swing 2.12** dependencies by inserting the following code in the `pom.xml` file:

```
<project>
  ...
  <dependencies>
    ...
    <dependency>
      <groupId>org.apache.spark</groupId>
      <artifactId>spark-core_2.12</artifactId>
      <version>3.4.0</version>
    </dependency>
    <dependency>
      <groupId>org.scala-lang.modules</groupId>
      <artifactId>scala-swing_2.12</artifactId>
      <version>3.0.0</version>
    </dependency>
  </dependencies>
</project>
```

- Add the **scala-maven-plugin** and **maven-assembly-plugin** build plugins by inserting the following code in the `pom.xml` file:

```
<project>
  ...
  <build>
    <plugins>
      <plugin>
        <groupId>net.alchim31.maven</groupId>
        <artifactId>scala-maven-plugin</artifactId>
        <version>3.2.2</version>
        <executions><execution><goals><goal>
          compile
        </goal></goals></execution></executions>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-assembly-plugin</artifactId>
        <version>3.3.0</version>
        <configuration><descriptorRefs><descriptorRef>
          jar-with-dependencies
        </descriptorRef></descriptorRefs></configuration>
        <executions><execution>
          <phase>package</phase>
          <goals><goal>single</goal></goals>
        </execution></executions>
      </plugin>
    </plugins>
  </build>
</project>
```

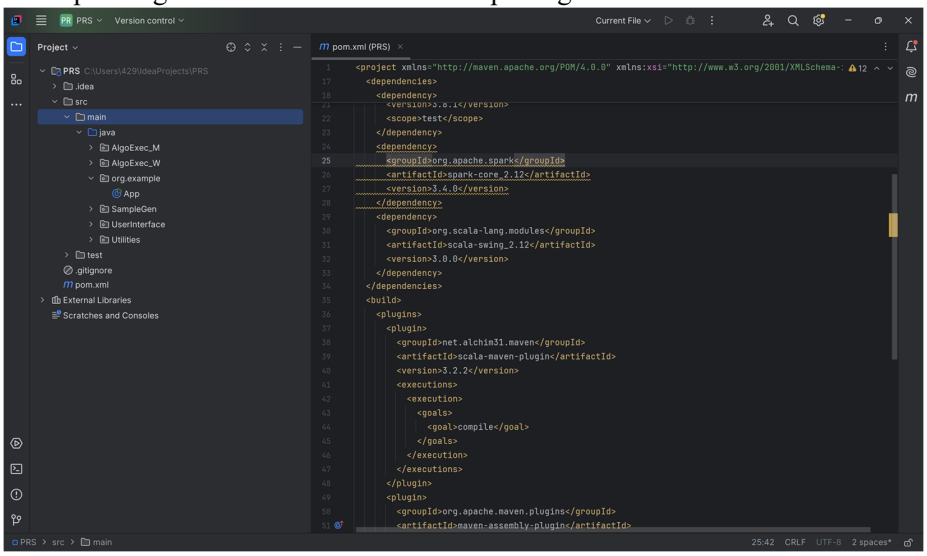
- Right-click on the blank area of the `pom.xml` file, then select **Maven → Reload Project** to reload the project.



3.1.2 Deploying the Source Code to the Maven Project

A. Moving Source Code to the Project

The source code for the Java version of PRS contains five subfolders: `SampleGen`, `AlgoExec_M`, `AlgoExec_W`, `UserInterface`, and `Utilities`. Users should create a package for each subfolder under the `/src/main/java` directory of the Maven project and move the corresponding source code files into these packages.



B. Replacing the `runSimulation(double[] argsJ,long seedJ)` Method

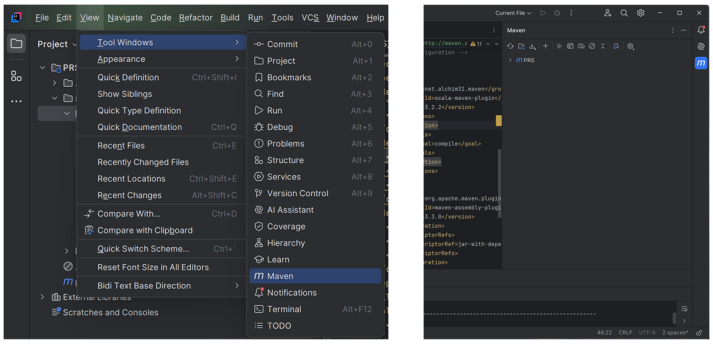
After deploying the code, users should modify the `runSimulation(double[] argsJ,long seedJ)` method within the `SampleGenerate` class, located in the `SampleGen` package. This method is responsible for generating simulation samples. Each time this method is invoked, a simulation sample is generated from an alternative, and the output is of the `double` data type. When using PRS to solve different problems, this method needs to be adjusted to properly generate simulation samples. When modifying the method, two rules must be followed: 1) To generate simulation samples, one needs to know the information of the alternative subject for simulation. The first argument of the method, i.e., `argsJ`, is an array of the `double`

data type. This array stores the information of the alternative subject for simulation in the same way as the information is stored in the `alternatives.txt` file. Whenever alternative information is needed, users can refer to this array; 2) To ensure that the results are replicable, we require users to set seed for the random number generator used within the method. The second argument of the method, i.e., `seedJ`, is a long value that designates the seed. Below is a demonstration of the method for the throughput maximization problem considered in the paper:

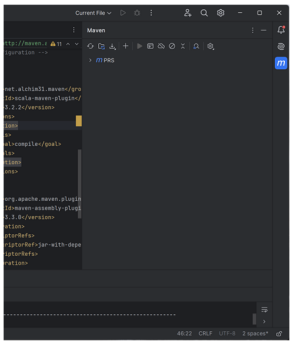
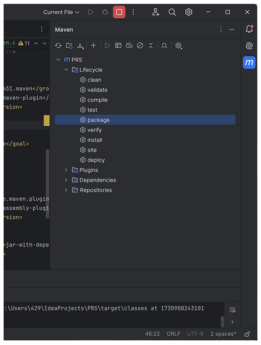
```
public double runSimulation(double[] argsJ,long seedJ) {
    double s1 = argsJ[1], s2 = argsJ[2], s3 = argsJ[3];
    double b2 = argsJ[4], b3 = argsJ[5];
    Random R = new Random(seedJ);
    // Main simulation process starts
    double[][] ST = new double[2050][3];
    double[][] ET = new double[2050][3];
    for ( int i = 0; i < 2050; i++){
        ST[i][0] = -Math.log(1-R.nextDouble())/s1;
        ST[i][1] = -Math.log(1-R.nextDouble())/s2;
        ST[i][2] = -Math.log(1-R.nextDouble())/s3;
    }
    ET[0][0] = ST[0][0];
    ET[0][1] = ET[0][0] + ST[0][1];
    ET[0][2] = ET[0][1] + ST[0][2];
    for ( int i = 1; i < 2050; i++){
        ET[i][0] = ET[i-1][0] + ST[i][0];
        ET[i][1] = Math.max(ET[i-1][1],ET[i][0]) + ST[i][1];
        ET[i][2] = Math.max(ET[i-1][2],ET[i][1]) + ST[i][2];
        if ( i >= b2 ) {
            ET[i][0] = Math.max(ET[i][0],ET[(int)(i-b2)][1]);
        }
        if ( i >= b3 ) {
            ET[i][1] = Math.max(ET[i][1],ET[(int)(i-b3)][2]);
        }
    }
    // Main simulation process ends
    return (2050 - 2000) / (ET[2050-1][2] - ET[2000-1][2]);
}
```

3.1.3 Packaging the Maven Project as a JAR file

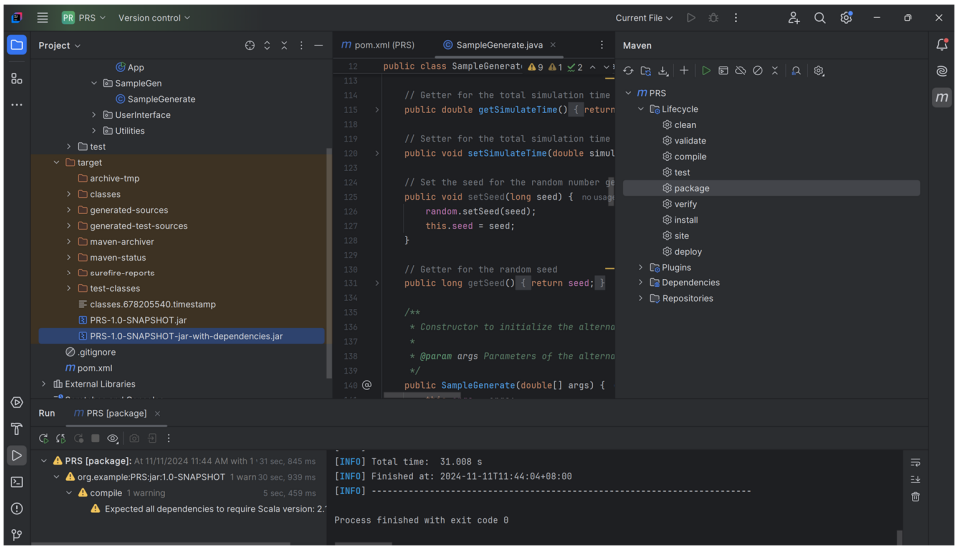
- Click **View → Tool Windows → Maven**. Then the Maven Tool Window opens.



- Expand **PRS** and the **Lifecycle** section.

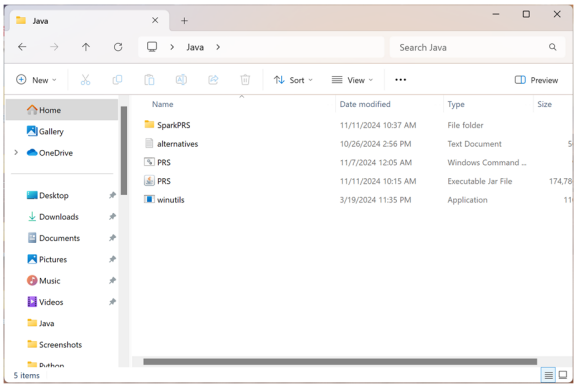
- Double-click the **package** goal to compile and package the project. Once the project is packaged, two JAR files will be generated in the target directory. The file with "`-with-dependencies`" in its name is the one to use, e.g., `PRS-1.0-SNAPSHOT-jar-with-dependencies.jar`. It is essential that users rename this JAR file to PRS.jar.



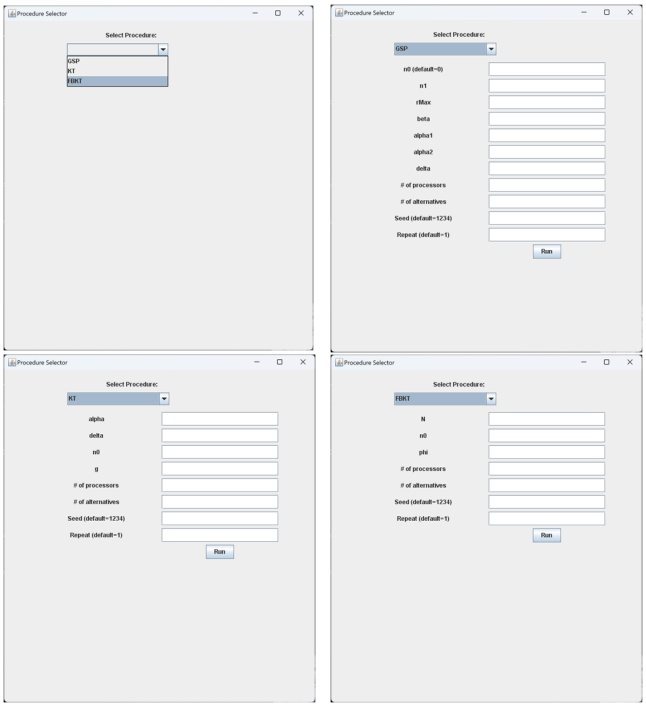
3.1.4 Running the Java Version of PRS

A. Running the Java version of PRS on Windows

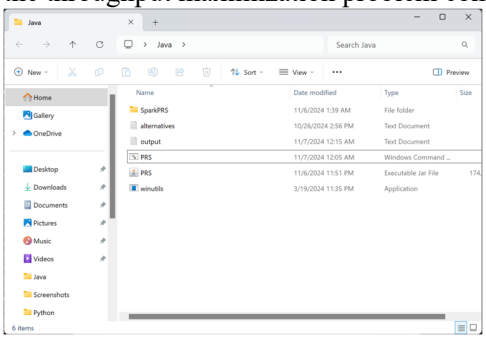
- Place the `PRS.cmd` file, the `winutils.exe` file, the `SparkPRS` folder, the `alternatives.txt` file, and the `PRS.jar` file in the same directory.

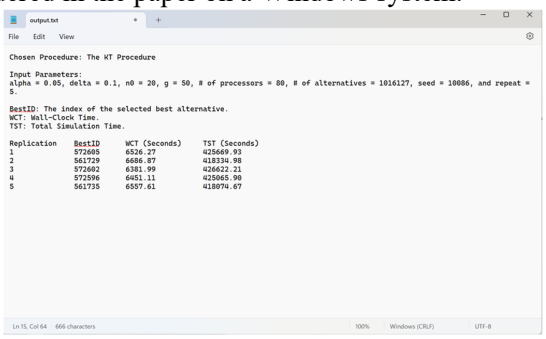


- Run the `PRS.cmd` file and select the **Java version of PRS**.
- Select the appropriate procedure, set values for the parameters accordingly, and click **Run**.



- Upon completion of the procedure, an `output.txt` file will be created in the working directory, i.e., where the `PRS.cmd` file locates, storing the corresponding experiment results. The figures below show the `output.txt` file obtained by using the KT procedure to solve the throughput maximization problem considered in the paper on a Windows system.





B. Running the Java version of PRS on Linux

When using PRS for the first time, users need to install Secure Shell (SSH) and configure passwordless SSH login. Please follow the installation and configuration steps outlined below:

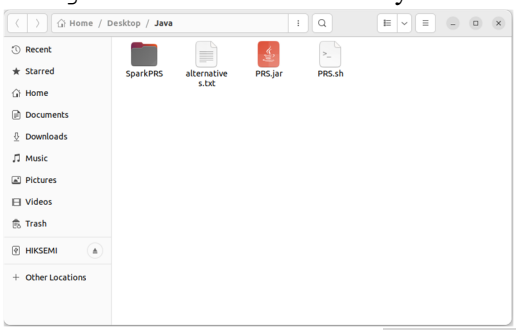
- Typically, the SSH client is pre-installed; however, users must also install the SSH server by executing the `sudo apt-get install openssh-server` command in the terminal.
- Once the installation is complete, users can verify the SSH server by logging into the local machine with the `ssh localhost` command. At this point, a prompt will appear (the SSH first login prompt). Type **yes** and press **Enter**. Follow the prompts to enter the password, which will log the user into the local machine. Then, users can execute the `exit` command to exit the SSH session.
- To enable passwordless login, users can execute the following commands in the terminal:
  `cd ~/.ssh`
  `ssh-keygen -t rsa`

```
cat ./id_rsa.pub >> ./authorized_keys
```

Notice that during the execution of the `ssh-keygen -t rsa` command, users may encounter prompts requiring input for several configurations. At each prompt, users can press **Enter** to skip the setting and proceed with the default option.

Then, users may run PRS by following the steps outlined below:

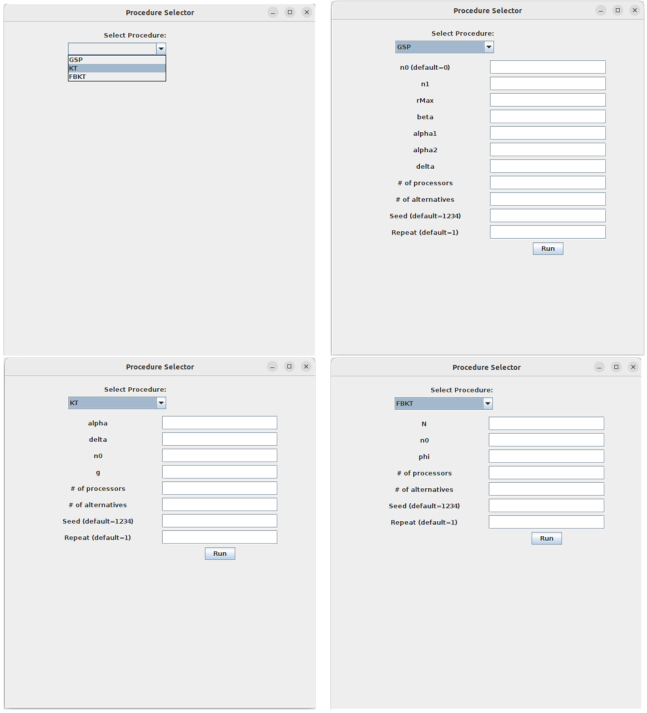
- Place the `PRS.sh` file, the `SparkPRS` folder, the `alternatives.txt` file, and the `PRS.jar` file in the same directory.



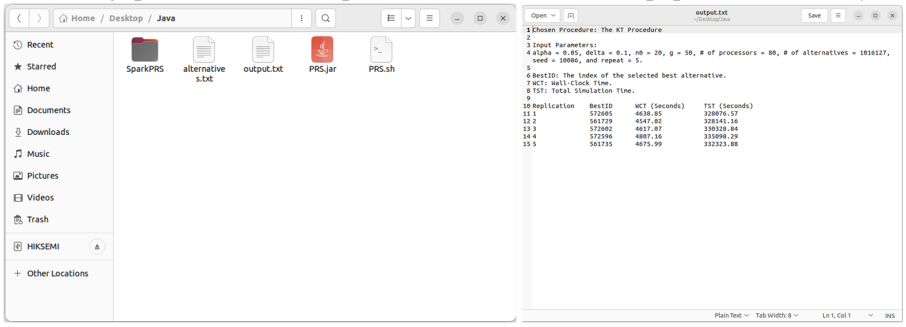
- Open the terminal and run the `cd /path/to/` command to navigate to the directory containing the `PRS.sh` file. Notice that users need to replace `/path/to` with the actual path to the directory where the `PRS.sh` file locates. For example, if the `PRS.sh` file locates in `~/Desktop/Java`, users should run `cd ~/Desktop/Java`
- Once inside the correct directory, execute the following commands to run the `PRS.sh` file and choose **Java**:

```
chmod +x PRS.sh
./PRS.sh
```

- Select the appropriate procedure, set values for the parameters accordingly, and click **Run**.



- Upon completion of the procedure, an `output.txt` file will be created in the working directory, i.e., where the `PRS.sh` file locates, storing the corresponding experiment results. The figures below show the `output.txt` file obtained by using the KT procedure to solve the throughput maximization problem considered in the paper on a Linux system.



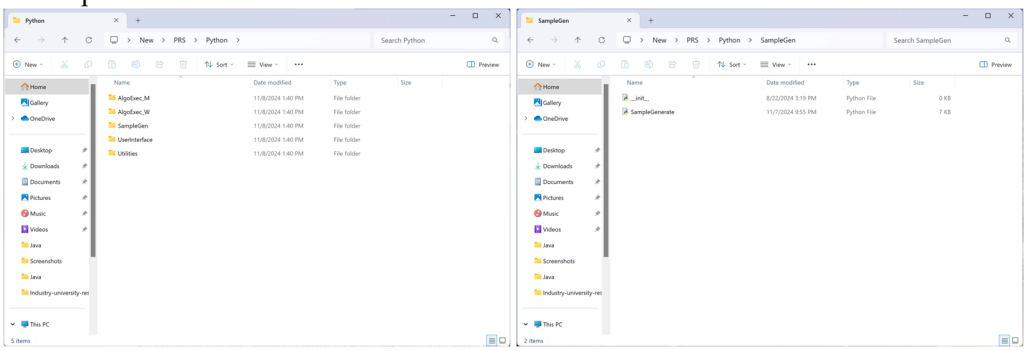
3.2 Setting up the Python Version of PRS

Compared to the Java version of PRS, the remaining setup process of the Python version of PRS is relatively simple. The remaining setup process for the Java version of PRS consists of two steps: 1) Editing the source code; 2) Running PRS.

3.2.1 Editing the Source Code

A. Replacing the `runSimulation(argsP,seedP)` Function

- The source code for the Python version of PRS also contains five subfolders: `SampleGen`, `AlgoExec_M`, `AlgoExec_W`, `UserInterface`, and `Utilities`. To set up the Python version of PRS, Users must modify the function which is responsible for generating simulation samples in the source code file. Specifically, the `runSimulation(argsP,seedP)` function within the `SampleGenerate.py` file, located in the `SampleGen` subfolder, needs to be updated.



- The `runSimulation(argsP,seedP)` function takes two arguments: `argsP` and `seedP`. The first argument is a NumPy array of floats that record the information of the alternative subject for simulation. The information is stored in the same manner as it is stored in the `alternatives.txt` file. The second argument is an integer that designates the seed for the random number generator used within the function.
- To make the modifications, users can open the `SampleGenerate.py` file using any text editor, such as Notepad, and update the function accordingly. Below is a demonstration of the function for the throughput maximization problem considered in the paper:

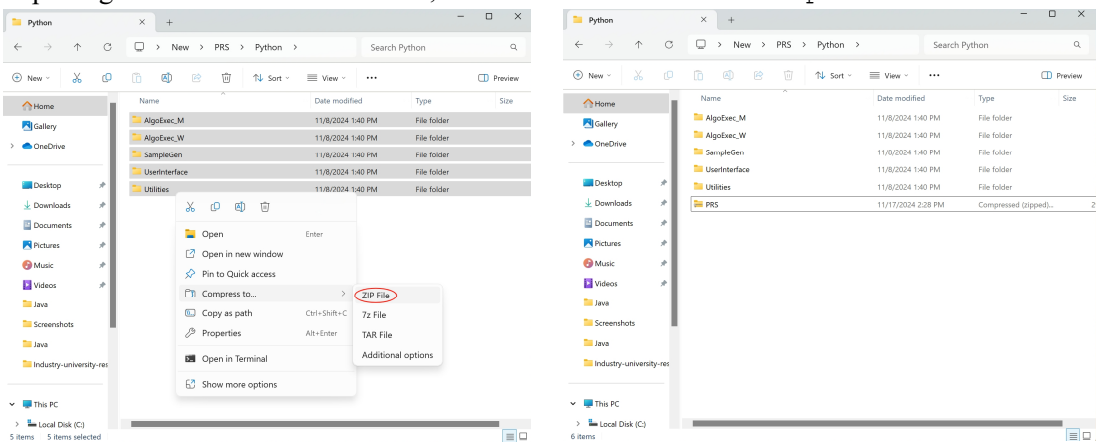
```
from numba import jit
import numpy as np
```

```python
@jit(nopython=True) # Applying JIT decorator (Optional)
def run_simulation(argsP,seedP):
    s1, s2, s3 = argsP[1], argsP[2], argsP[3]
    b2, b3 = argsP[4], argsP[5]
    np.random.seed(seedP)
    # Main simulation process starts
    ST = np.zeros((2050, 3))
    ET = np.zeros((2050, 3))
    for i in range(2050):
        ST[i, 0] = np.random.exponential(1 / s1)
        ST[i, 1] = np.random.exponential(1 / s2)
        ST[i, 2] = np.random.exponential(1 / s3)
    ET[0, 0] = ST[0, 0]
    ET[0, 1] = ET[0, 0] + ST[0, 1]
    ET[0, 2] = ET[0, 1] + ST[0, 2]
    for i in range(1, 2050):
        ET[i, 0] = ET[i - 1, 0] + ST[i, 0]
        ET[i, 1] = max(ET[i - 1, 1], ET[i, 0]) + ST[i, 1]
        ET[i, 2] = max(ET[i - 1, 2], ET[i, 1]) + ST[i, 2]
        if i >= b2:
            ET[i, 0] = max(ET[i, 0], ET[int(i - b2), 1])
        if i >= b3:
            ET[i, 1] = max(ET[i, 1], ET[int(i - b3), 2])
    # Main simulation process ends
    return (2050 - 2000) / (ET[-1, 2] - ET[2000 - 1, 2])
```

Notice that the `@jit(nonpython=True)` in the third line of the code is used to apply the JIT decorator to accelerate the execution of the `run_simulation(argsP, seedP)` function. In some cases, this technique may not be applicable, and users should remove this line when developing their own `simulation(argsP, seedP)` function in these cases.

B. Compressing the Source Code Files

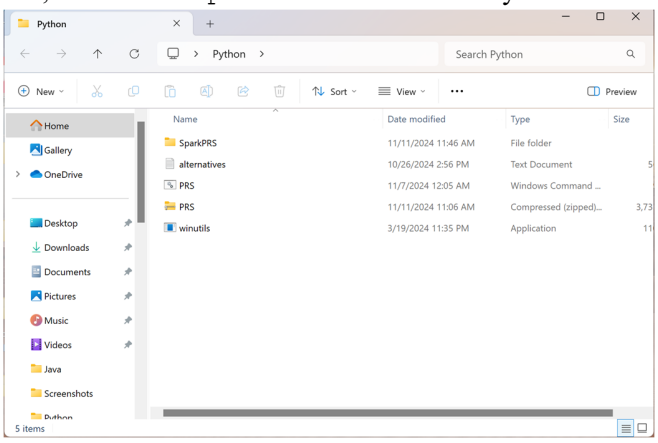
After modifying `runSimulation(argsP, seedP)`, select the five subfolders: `SampleGen`, `AlgoExec_M`, `AlgoExec_W`, `UserInterface`, and `Utilities`, right-click, and choose to package them into a ZIP file. Then, rename the ZIP file as `PRS.zip`.



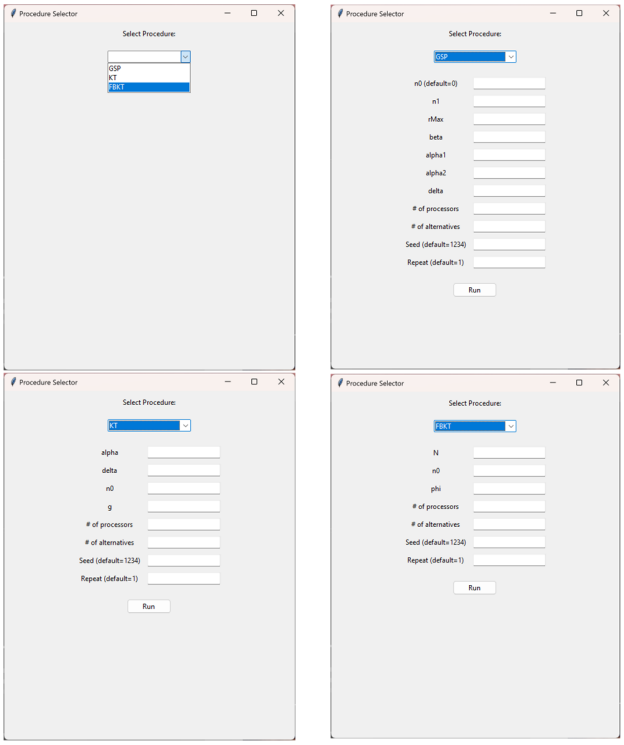
3.2.2 Running the Python version of PRS

A. Running the Python version of PRS on Windows

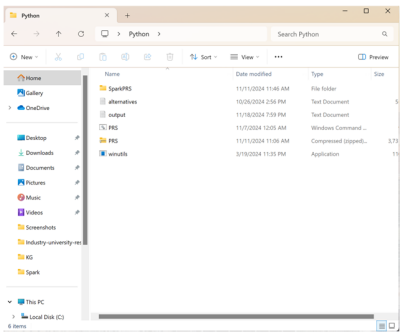
- Place the `winutils.exe` file, `PRS.cmd` file, `SparkPRS` folder, `alternatives.txt` file, and `PRS.zip` file in the same directory.

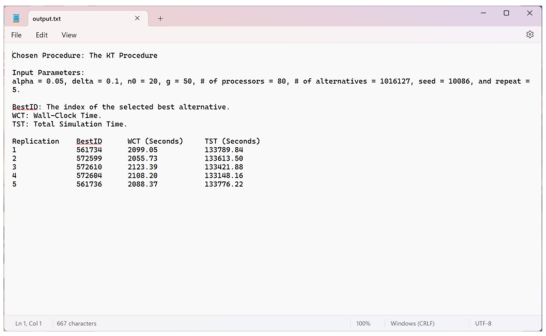


- Run the `PRS.cmd` file and select the **Python version of PRS**.
- Select the appropriate procedure as needed, set values for the parameters accordingly, and click **Run** to start.



- Upon completion of the procedure, an `output.txt` file will be created in the working directory, i.e., where the `PRS.cmd` file locates, storing the corresponding experiment results. The figures below show the `output.txt` file obtained by using the KT procedure to solve the throughput maximization problem considered in the paper on a Windows system.





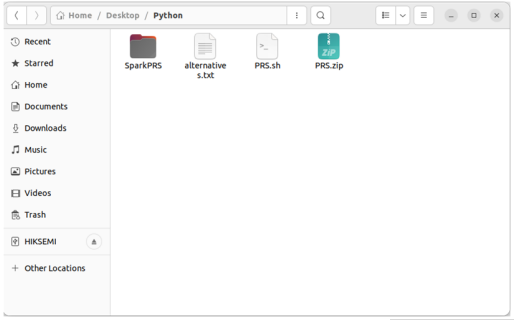
B. Running the Python version of PRS on Linux

When using PRS for the first time, users need to install Secure Shell (SSH) and configure passwordless SSH login. Please follow the installation and configuration steps outlined below:

- Typically, the SSH client is pre-installed; however, users must also install the SSH server by executing the `sudo apt-get install openssh-server` command in the terminal.
- Once the installation is complete, users can verify the SSH server by logging into the local machine with the `ssh localhost` command. At this point, a prompt will appear (the SSH first login prompt). Type **yes** and press **Enter**. Follow the prompts to enter the password, which will log the user into the local machine. Then users can execute the `exit` command to exit the SSH session.
- To enable passwordless login, users can execute the following commands in the terminal:
  ```
  cd ~/.ssh
  ssh-keygen -t rsa
  cat ./id_rsa.pub >> ./authorized_keys
  ```
  Notice that during the execution of the `ssh-keygen -t rsa` command, users may encounter prompts requiring input for several configurations. At each prompt, users can press **Enter** to skip the setting and proceed with the default option.

Then, users may run PRS by following the steps outlined below:

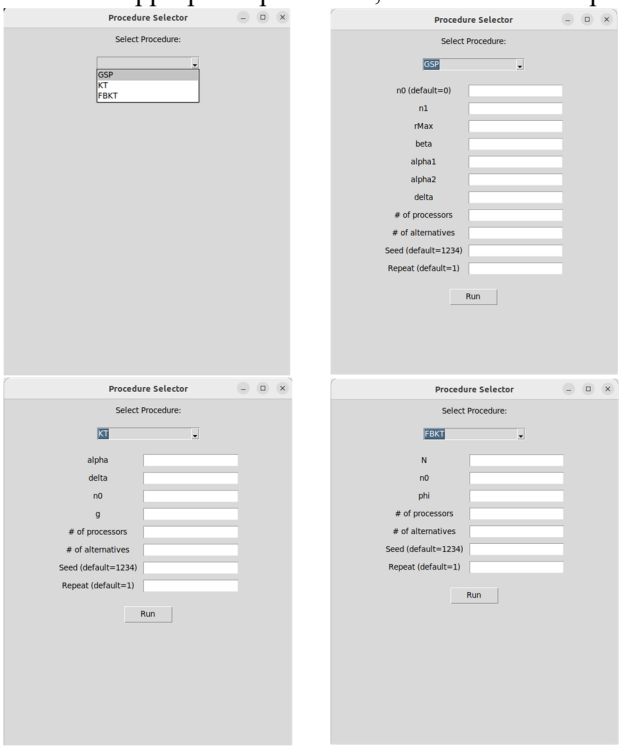
- Place the `PRS.sh` file, the `SparkPRS` folder, the `alternatives.txt` file, and the `PRS.zip` file in the same directory.



- Open the terminal and run the `cd /path/to/` command to navigate to the directory containing the `PRS.sh` file. Notice that users need to replace `/path/to` with the actual path to the directory where the `PRS.sh` file locates. For example, if the `PRS.sh` file locates in `~/Desktop/Python`, users should run `cd ~/Desktop/Python`
- Once inside the correct directory, execute the following commands to run the `PRS.sh` file and choose **Python**:
  ```
  chmod +x PRS.sh
  ```

`./PRS.sh`

- Select the appropriate procedure, set values for the parameters accordingly, and click **Run**.



- Upon completion of the procedure, an `output.txt` file will be created in the working directory, i.e., where the `PRS.sh` file locates, storing the corresponding experiment results. The figures below show the `output.txt` file obtained by using the KT procedure to solve the throughput maximization problem considered in the paper on a Linux system.